PSEUDO-CODE

We present basic pseudo-code for some of the algorithms, discussed in the Steps. In our experience, students do benefit by studying the pseudo-code of a method at the same time as they learn it in a Step. If they are familiar with a programming language, they should attempt to convert at least some of the pseudo-codes into computer programs, and apply them to the set Exercises.

Bisection Method (STEP 7)

The equation is *f (x)* = 0.

```
1    read a, b, ε
2    repeat
3      x = (a + b)/2
4      if f(x) = 0 then do:
5        print 'Root is', x
6        stop
7      endif
8      if f(a) * f(x) > 0 then do:
9        a = x
10     else do:
11       b = x
12     endif
13   until b − a < ε
14   print 'Approximation to root is', x
```

Points for study:

1. What is the input used for?
2. Explain the purpose of Lines 8 - 12.
3. Amend the speudo-code, so that the process will always stop after preset *M* iterations.
4. Amend the pseudo-code so that the process will stop as soon as $|f(x)| < \epsilon$.
5. Write a computer program, based on this speudo-code.
6. Use your program to solve Exercises **1** and **2** in the Applied Exercises.

- **Method of False position (STEP 8)**

The equation is *f (x)* = 0.

```
1   read a, b, ϵ
2   repeat
3     x = (a * f(b) − b * f(a))/(f(b) − f(a))
4     if f(x) = 0 then do:
5       print 'Root is', x
6       stop
7     endif
8     if f(a) * f(x) > 0 then do:
9       a = x
10    else do:
11      b = x
12    endif
13  until |f(x)| < ϵ
14  print 'Approximation to root is', x
```

Points for study

1. What are the input values used for?
2. Under what circumstances may the process stop with a large error in *x*?
3. Amend the pseudo-code so that the process will stop after *M* iterations, if the condition in Line 13 is not satisfied.
4. Write a computer Program based on the pseudo-code.
5. Use your program to solve Exercises **1** and **2** in the Applied Exercises.

- **Newton-Raphson iterative method (STEP 10)**

The equation is *f (x)* = 0.

```
read a, M, ϵ
N = 0
repeat
  δ = f(a)/f'(a)
  a = a − δ
  N = N + 1
until |δ| < ϵ or N = M
print 'Approximation to root is', a
if |δ| ≥ ϵ then do:
  print 'required accuracy not reached in', M, 'iterations'
endif
```

Points for study

8

1. How are the input values used?
2. Why is *M* given in the output of Line 10?
3. What happens if *f'(a)* is very small?
4. Amend the pseudo-code to take suitable action if *f'(a)* is very small.
5. Write a computer program based on the pseudo-code.
6. Use your program to solve Exercises **1** and **2** in the Applied Exercises.

- **Gauss Elimination (STEP 11)**

The system is:

$$
\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n
\end{array}
$$

```
1   read n, a_11, ..., a_nn, b_1, ..., b_n
2   for k = 1 to n - 1 do:
3     for i = k + 1 to n do:
4       m = a_ik / a_kk
5       for j = k + 1 to n do:
6         a_ij = a_ij - m * a_kj
7       endfor
8       b_i = b_i - m * b_k
9     endfor
10  endfor
11  x_n = b_n / a_nn
12  for i = n - 1 downto 1 do:
13    x_i = b_i
14    for j = i + 1 to n do:
15      x_i = x_i - a_ij * x_j
16    endfor
17    x_i = x_i / a_ii
18  endfor
19  print 'Approximate solution is', x_1, ..., x_n
```

Points for study

1. Explain what happens in Lines 2 - 10.
2. What process is implemented in Lines 11 - 18`?
3. Amend the pseudo-code so that the program terminates with an informative message when a zero pivot element is

found.
4. Write a program based on the pseudo-code.
5. Use your program to solve Exercises **3** and **4** in the Applied Exercises.

- **Gauss-Seidel Iteration(STEP 13)**

The system is:

$$
\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n
\end{array}
$$

```
1   read n, a11, ..., ann, b1, ..., bn
2   for k = 1 to n - 1 do:
3     for i = k + 1 to n do:
4       m = aik/akk
5       for j = k + 1 to n do:
6         aij = aij - m * akj
7       endfor
8       bi = bi - m * bk
9     endfor
10  endfor
11  xn = bn/ann
12  for i = n - 1 downto 1 do:
13    xi = bi
14    for j = i + 1 to n do:
15      xi = xi - aij * xj
16    endfor
17    xi = xi/aii
18  endfor
19  print 'Approximate solution is', x1, ..., xn
```

Points for study

1. What is the purpose of the number s?
2. What are the $y_1, y_2\ 1, \ldots, y_n$ used for?
3. Why is it possible to replace the y_j in Line 13 by x_j?
4. Amend the pseudo-code to allow a maximum of M iterations.
5. Write a program based on the pseudo-code.
6. Use thc computer program to solve the system:

$$\begin{aligned} 8x + y - 2z &= 5 \\ x - 7y + z &= 9 \\ 2x + 9z &= 11 \end{aligned}$$

7. Use your program to solve Exercises **3** and **4** in the Applied Exercises.

- **Newton divided difference formula (STEP 24)**

You are to calculate for given data $x_0, x_1, \ldots, x_n, f(x_0), f(x_1), \ldots, f(x_n)$, and given $x \in [x_0, x_n]$, the interpolating polynomial $P_n(x>$ of degree n. (The algorithm is based on divided differences.)

```
1   read n, x, x0, ..., xn, f(x0), ..., f(xn)
2   for i = 0 to n do:
3     d_{i,0} = f(x_i)
4   endfor
5   for i = 1 to n do:
6     for j = 1 to i do:
7       d_{i,j} = (d_{i,j-1} - d_{i-1,j-1})/(x_i - x_{i-j})
8     endfor
9   endfor
10  sum = d_{0,0}
11  prod = 1.0
12  for i = 1 to n do:
13    prod = prod * (x - x_{i-1})
14    sum = sum + d_{i,i} * prod
15  endfor
16  print 'Approximation at x =', x, 'is', sum
```

Points for study

1. Follow the pseudo-code through with the data $n = 2$, $.x = 1.5$, $x_0 = 0$, $f(.x_0) = 2.5$,$.x_1 = 1$, $f(x_1) = 4.7$, xSS $_2 = 3$, and f $x_2) = 3.1$. Verify that the values d_{ii} calculated are the divided differences $f(x_0, \ldots, .x_I)$.
2. What quantity (in algebraic terms) is calculated in Lines 10 - 15?
3. Amend the pseudo-code so that the values $P_1(x)$, $P_2(x)$P(x), . . ., $P_{n\text{-}1}(x)$are also printed out.
4. Write a computer program based on the pseudo-code.
5. Use your program to estimate $f(2)$ for the data given in **1** above.
6. For the data, given in Exercise **6** of the Applied Exercises,

use the program to obtain an estimate of $J_0(0.25)$.

- **Trapezoidal Rule([STEP 30]())**

The integral is:

$$\int_a^b f(x)\,dx.$$

```
1   read a, b, N, M, ε
2   done = false
3   U = 0.0
4   repeat
5     h = (b − a)/N
6     s = (f(a) + f(b))/2
7     for i = 1 to N − 1 do:
8       x = a + i ∗ h
9       s = s + f(x)
10    endfor
11    T = h ∗ s
12    if |T − U| < ε then do:
13      done = true
14    else do:
15      N = 2 ∗ N
16      U = T
17    endif
18  until N > M or done
19  print 'Approximation to integral is', T
20  if N > M then do:
21    print 'required accuracy not reached with M =', M
22  endif
```

Points for study

1. What are the input values used for?
2. What value (in algebraic terms) does T have after Line 11?
3. What is the purpose of Lines 12-17?
4. Write a program based on the pseudo-code.
5. Apply your program to Exercises 7 and **8** of the [Applied Exercises]().

- **Gauss integration formula ([STEP 32]())**

The integral is:

$\int_a^b f(x)\,dx.$

Use the Gauss two-point formula.

```
1  read a, b
2  x1 = (b + a − (b − a)/√3)/2
3  x2 = (b + a + (b − a)/√3)/2
4  I = (b − a) * (f(x1) + f(x2))/2
5  print 'Approximation to integral is', I
```

Points for study

1. What is the purpose of Lines 2 and 3?
2. What changes are required to produce an algorithm based on the Gauss three-point formula?
3. Write a computer program based on this pseudo-code.
4. Use your program to solve Exercises 7 and **8** of the Applied Exercises.

5. **Runge-Kutta method (STEP 33)**

   Process the equation $y' = f(x,y)$ and use the usual fourth-order method.

```
read x, y, h, M
print x, y
N = 0
repeat
   k1 = h * f(x, y)
   x = x + h/2
   z = y + k1/2
   k2 = h * f(x, z)
   z = y + k2/2
   k3 = h * f(x, z)
   x = x + h/2
   z = y + k3
   k4 = h * f(x, z)
   y = y + (k1 + 2k2 + 2k3 + k4)/6
   print x, y
   N = N + 1
until N = M
```

Points for study

1. What are the input values used for?

2. How many times is the function *f* evaluated between Lines 4 and 17?
3. Amend the pseudo-code for use with the second-order Runge-Kutta method.
4. Write a computer program based on the pseudo-code.
5. Use the computer program to solve Exercises **9** and **10** of the Applied Exercises.